Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

Client experimental GoDriveCarBox

Cod sursa sistem achizitie si analiza

GDCB Explorer

Proiect	GoDrive
Beneficiar	GODRIVE SRL
Contract	Nr. 2/25.11.2016
Data modificare	2017.05.29
Data creere	2017.02.02
Versiune	1.2.3.1
Descriere	Implementare modele predictie defecte autovehicule

Contents

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

Introducere

Prezentul document descrie motorul de achizitie si management al datelor aferente proiectului GoDrive CarBox. Intreaga structura de date, cod sursa, fisiere de configurare se poate regasi in format GIT la adresa:

https://github.com/orgs/GoDriveCarBox/dashboard

In urmatoarele capitole sunt prezentate urmatoarele livrabile disponibile in repository-ul GIT mentionat anterior:

1. Structura si configurarea sistemului prin utilizarea fisierelor text JSON
2. Structura si definitia datelor pentru relationarea cu clientii GoDrive CarBox
3. Structura, definitia si exemple de date telemetrice generate
4. Codul sursa al modulului server Python (GDCB Explorer)
5. Codul sursa al modulului de machine learning in versiunea sa 2 (fata de livrabilul anterior 2.1) realizat in limbaj C++ portabil si utilizand bibliotecile de calcul numeric paralel Eigen.

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

Structura Datelor

Structura configurarii sistemului

Intregul sistem este configurabil prin modificarea fisierelor de tip text JSON care contine absolut toate informatiile necesare si suficiente in vederea configurarii complete a sistemului GDCB Explorer. Mai jos sunt prezentate fisierele text standard de configurare a experimentul la data pregatirii acestui document livrabil:

```
{
     "PREDICTOR_TABLE" : "codes_v1",
     "CARS_TABLE" : "Cars",
     "ACCOUNTS_TABLE" : "Accounts",
     "RAWDATA_TABLE" : "RawData",
     "CODE_FIELD" : "ID",
     "SIZE_FIELD": "Size",

     "VIEW_ALLDATA": "vw_getdata",

     "RAW_IGNR_FIELD": "ID",
     "RAW_NVAL_FIELD": "IntValue",
     "RAW_SVAL_FIELD": "StrValue",
     "RAW_CODE_FIELD": "CodeID",
     "RAW_TIME_FIELD": "TimeStamp",
     "RAW_CARI_FIELD": "CarID",

     "DEVICE_SERVICE": "DevServiceTest0",
     "DEVICE_PROTOCOL": "DevServiceTest0_Proto"
```

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

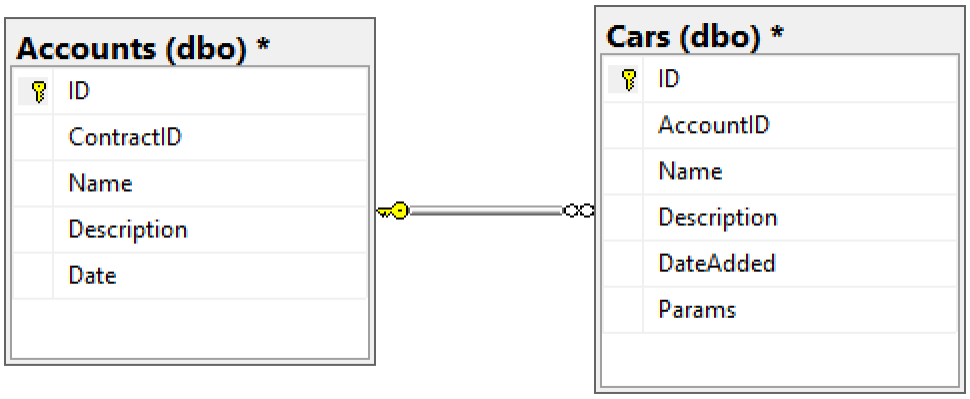
+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
}

{
      "server" : "carbox.database.windows.net",
      "database" : "Carbox",
      "username" : "carbox@carbox",
      "password" : "GDCBnpsf0517",
      "driver" : "{ODBC Driver 13 for SQL Server}",
      "datafolder" : "d:/GoogleDrive/_godrive_data"
}
```

Structura de date la nivel relationare cu clientii:




```
USE [Carbox]
GO

/****** Object:  Table [dbo].[Accounts]    Script Date: 2017-05-29 18:05:36 ******/
DROP TABLE [dbo].[Accounts]
GO

/****** Object:  Table [dbo].[Accounts]    Script Date: 2017-05-29 18:05:36 ******/
SET ANSI_NULLS ON
```

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

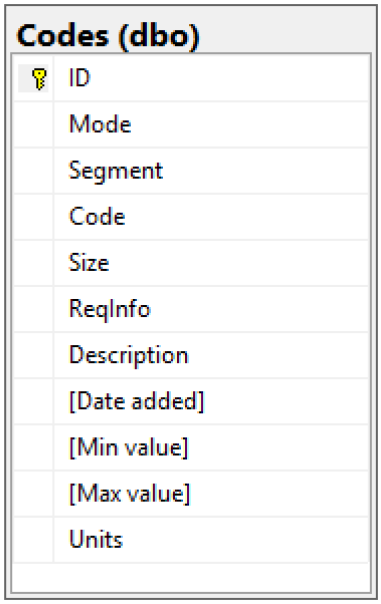
Website:
http://www.4esoft.com

email: office@4esoft.com

```
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Accounts](
	[ID] [bigint] IDENTITY(1,1) NOT NULL,
	[ContractID] [bigint] NULL,
	[Name] [varchar](255) NULL,
	[Description] [varchar](255) NULL,
	[Date] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
	[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
)
GO

USE [Carbox]
GO
/****** Object:  Table [dbo].[Cars]    Script Date: 2017-05-29 18:05:09 ******/
DROP TABLE [dbo].[Cars]
GO
/****** Object:  Table [dbo].[Cars]    Script Date: 2017-05-29 18:05:10 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Cars](
	[ID] [bigint] IDENTITY(1,1) NOT NULL,
	[AccountID] [bigint] NULL,
	[Name] [varchar](255) NULL,
	[Description] [varchar](255) NULL,
	[DateAdded] [datetime] NULL,
	[Params] [varchar](255) NULL,
PRIMARY KEY CLUSTERED
(
	[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
)
GO
```

Structura de date la nivel de achizitiei de date



Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
USE [Carbox]
GO

/****** Object:  Table [dbo].[Codes]    Script Date: 2017-05-29 18:06:09 ******/
DROP TABLE [dbo].[Codes]
GO

/****** Object:  Table [dbo].[Codes]    Script Date: 2017-05-29 18:06:09 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Codes](
	[ID] [bigint] NOT NULL,
	[Mode] [varchar](max) NULL,
	[Segment] [varchar](max) NULL,
	[Code] [varchar](max) NULL,
	[Size] [float] NULL,
	[ReqInfo] [float] NULL,
	[Description] [varchar](max) NULL,
	[Date added] [varchar](max) NULL,
	[Min value] [varchar](max) NULL,
	[Max value] [varchar](max) NULL,
	[Units] [varchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[ID] ASC
```

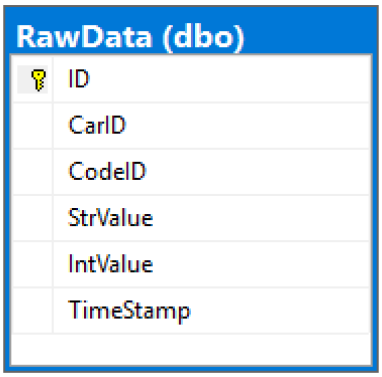
Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
)

GO
```




```
USE [Carbox]
GO

/****** Object:  Table [dbo].[RawData]    Script Date: 2017-05-29 18:06:56 ******/
DROP TABLE [dbo].[RawData]
GO

/****** Object:  Table [dbo].[RawData]    Script Date: 2017-05-29 18:06:56 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[RawData](
```

```
        [ID] [bigint] IDENTITY(1,1) NOT NULL,
        [CarID] [bigint] NOT NULL,
        [CodeID] [bigint] NULL,
        [StrValue] [varchar](255) NULL,
        [IntValue] [bigint] NULL,
        [TimeStamp] [datetime] NULL,
PRIMARY KEY CLUSTERED
(
        [ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
)

GO
```

Structura datelor de telemetrie

Prezentarea codurilor de telemetrie

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1000	Mode 01	00-20	0	4	NULL	PIDs supported range 00-20	2017-05-26 20:20	NULL	NULL	NULL
1001	Mode 01	00-20	1	4	NULL	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)	2017-05-26 20:20	NULL	NULL	NULL
1002	Mode 01	00-20	2	2	NULL	Freeze DTC	2017-05-26 20:20	NULL	NULL	NULL

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1003	Mode 01	00-20	3	2	NULL	Fuel system status	2017-05-26 20:20	NULL	NULL	NULL
1004	Mode 01	00-20	4	1	NULL	Calculated engine load	2017-05-26 20:20	0	100	%
1005	Mode 01	00-20	5	1	NULL	Engine coolant temperature	2017-05-26 20:20	-40	215	deg.C
1006	Mode 01	00-20	6	1	NULL	Short term fuel trim?Bank 1	2017-05-26 20:20	-100 (Reduce Fuel: Too Rich)	99.2 (Add Fuel: Too Lean)	%
1007	Mode 01	00-20	7	1	NULL	Long term fuel trim?Bank 1	2017-05-26 20:20	NULL	NULL	NULL
1008	Mode 01	00-20	8	1	NULL	Short term fuel trim?Bank 2	2017-05-26 20:20	NULL	NULL	NULL
1009	Mode 01	00-20	9	1	NULL	Long term fuel trim?Bank 2	2017-05-26 20:20	NULL	NULL	NULL
1010	Mode 01	00-20	0A	1	NULL	Fuel pressure (gauge pressure)	2017-05-26 20:20	0	765	kPa
1011	Mode 01	00-20	0B	1	NULL	Intake manifold absolute pressure	2017-05-26 20:20	0	255	kPa
1012	Mode 01	00-20	0C	2	NULL	Engine RPM	2017-05-26 20:20	0	16383.75	rpm
1013	Mode 01	00-20	0D	1	NULL	Vehicle speed	2017-05-26 20:20	0	255	km/h
1014	Mode 01	00-20	0E	1	NULL	Timing advance	2017-05-26 20:20	-64	63.5	deg. before TDC
1015	Mode 01	00-20	0F	1	NULL	Intake air temperature	2017-05-26 20:20	-40	215	NULL
1016	Mode 01	00-20	10	2	NULL	MAF air flow rate	2017-05-26 20:20	0	655.35	deg.C
1017	Mode 01	00-20	11	1	NULL	Throttle position	2017-05-26 20:20	0	100	grams/sec
1018	Mode 01	00-20	12	1	NULL	Commanded secondary air status	2017-05-26 20:20	NULL	NULL	%
1019	Mode 01	00-20	13	1	NULL	Oxygen sensors present (in 2 banks)	2017-05-26 20:20	NULL	NULL	NULL
1020	Mode 01	00-20	14	2	NULL	Oxygen Sensor 1 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	0 / -100	1.275 / 99.2	NULL
1021	Mode 01	00-20	15	2	NULL	Oxygen Sensor 2 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	NULL	NULL	volts / %
1022	Mode 01	00-20	16	2	NULL	Oxygen Sensor 3 / A: Voltage /	2017-05-26 20:20	NULL	NULL	NULL

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
						B: Short term fuel trim				
1023	Mode 01	00-20	17	2	NULL	Oxygen Sensor 4 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	NULL	NULL	NULL
1024	Mode 01	00-20	18	2	NULL	Oxygen Sensor 5 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	NULL	NULL	NULL
1025	Mode 01	00-20	19	2	NULL	Oxygen Sensor 6 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	NULL	NULL	NULL
1026	Mode 01	00-20	1A	2	NULL	Oxygen Sensor 7 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	NULL	NULL	NULL
1027	Mode 01	00-20	1B	2	NULL	Oxygen Sensor 8 / A: Voltage / B: Short term fuel trim	2017-05-26 20:20	NULL	NULL	NULL
1028	Mode 01	00-20	1C	1	NULL	OBD standards this vehicle conforms to	2017-05-26 20:20	NULL	NULL	NULL
1029	Mode 01	00-20	1D	1	NULL	Oxygen sensors present (in 4 banks)	2017-05-26 20:20	NULL	NULL	NULL
1030	Mode 01	00-20	1E	1	NULL	Auxiliary input status	2017-05-26 20:20	NULL	NULL	NULL
1031	Mode 01	00-20	1F	2	NULL	Run time since engine start	2017-05-26 20:20	0	65535	NULL
1032	Mode 01	21-40	20	4	NULL	PIDs supported range 21-40	2017-05-26 20:20	NULL	NULL	seconds
1033	Mode 01	21-40	21	2	NULL	Distance traveled with malfunction indicator lamp (MIL) on	2017-05-26 20:20	0	65535	NULL
1034	Mode 01	21-40	22	2	NULL	Fuel Rail Pressure (relative to manifold vacuum)	2017-05-26 20:20	0	5177.27	km
1035	Mode 01	21-40	23	2	NULL	Fuel Rail Gauge Pressure (diesel- or gasoline direct injection)	2017-05-26 20:20	0	655350	kPa
1036	Mode 01	21-40	24	4	NULL	Oxygen Sensor 1 / AB: Fuel?Air Equivalence	2017-05-26 20:20	0 / 0	< 2 / < 8	kPa

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1037	Mode 01	21-40	25	4	NULL	Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	ratio / V
1038	Mode 01	21-40	26	4	NULL	Oxygen Sensor 2 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1039	Mode 01	21-40	27	4	NULL	Oxygen Sensor 3 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1040	Mode 01	21-40	28	4	NULL	Oxygen Sensor 4 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1041	Mode 01	21-40	29	4	NULL	Oxygen Sensor 5 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1042	Mode 01	21-40	2A	4	NULL	Oxygen Sensor 6 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1043	Mode 01	21-40	2B	4	NULL	Oxygen Sensor 7 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1044	Mode 01	21-40	2C	1	NULL	Oxygen Sensor 8 / AB: Fuel?Air Equivalence Ratio / CD: Voltage	2017-05-26 20:20	NULL	NULL	NULL
1045	Mode 01	21-40	2D	1	NULL	Commanded EGR	2017-05-26 20:20	0	100	NULL
1046	Mode 01	21-40	2E	1	NULL	EGR Error	2017-05-26 20:20	-100	99.2	%
1047	Mode 01	21-40	2F	1	NULL	Commanded evaporative purge	2017-05-26 20:20	0	100	%
1048	Mode 01	21-40	30	1	NULL	Fuel Tank Level Input	2017-05-26 20:20	0	100	%
1049	Mode 01	21-40	31	2	NULL	Warm-ups since codes cleared	2017-05-26 20:20	0	255	%
1050	Mode 01	21-40	32	2	NULL	Distance traveled since codes cleared	2017-05-26 20:20	0	65535	count
						Evap. System Vapor Pressure	2017-05-26 20:20	-8192	8191.75	km

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1051	Mode 01	21-40	33	1	NULL	Absolute Barometric Pressure	2017-05-26 20:20	0	255	Pa
1052	Mode 01	21-40	34	4	NULL	Oxygen Sensor 1 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	0 / -128	< 2 / <128	kPa
1053	Mode 01	21-40	35	4	NULL	Oxygen Sensor 2 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	ratio / mA
1054	Mode 01	21-40	36	4	NULL	Oxygen Sensor 3 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	NULL
1055	Mode 01	21-40	37	4	NULL	Oxygen Sensor 4 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	NULL
1056	Mode 01	21-40	38	4	NULL	Oxygen Sensor 5 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	NULL
1057	Mode 01	21-40	39	4	NULL	Oxygen Sensor 6 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	NULL
1058	Mode 01	21-40	3A	4	NULL	Oxygen Sensor 7 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	NULL
1059	Mode 01	21-40	3B	4	NULL	Oxygen Sensor 8 / AB: Fuel?Air Equivalence Ratio / CD: Current	2017-05-26 20:20	NULL	NULL	NULL
1060	Mode 01	21-40	3C	2	NULL	Catalyst Temperature: Bank 1- Sensor 1	2017-05-26 20:20	-40	6513.5	NULL
1061	Mode 01	21-40	3D	2	NULL	Catalyst Temperature: Bank 2- Sensor 1	2017-05-26 20:20	NULL	NULL	deg.C
1062	Mode 01	21-40	3E	2	NULL	Catalyst Temperature:	2017-05-26 20:20	NULL	NULL	NULL

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1063	Mode 01	21-40	3F	2	NULL	Bank 1- Sensor 2 Catalyst Temperature: Bank 2- Sensor 2	2017-05-26 20:20	NULL	NULL	NULL
1064	Mode 01	41-60	40	4	NULL	PIDs supported range 41-60	2017-05-26 20:20	NULL	NULL	NULL
1065	Mode 01	41-60	41	4	NULL	Monitor status this drive cycle	2017-05-26 20:20	NULL	NULL	NULL
1066	Mode 01	41-60	42	2	NULL	Control module voltage	2017-05-26 20:20	0	65.54	NULL
1067	Mode 01	41-60	43	2	NULL	Absolute load value	2017-05-26 20:20	0	25700	V
1068	Mode 01	41-60	44	2	NULL	Fuel?Air commanded equivalence ratio	2017-05-26 20:20	0	< 2	%
1069	Mode 01	41-60	45	1	NULL	Relative throttle position	2017-05-26 20:20	0	100	ratio
1070	Mode 01	41-60	46	1	NULL	Ambient air temperature	2017-05-26 20:20	-40	215	%
1071	Mode 01	41-60	47	1	NULL	Absolute throttle position B	2017-05-26 20:20	0	100	deg.C
1072	Mode 01	41-60	48	1	NULL	Absolute throttle position C	2017-05-26 20:20	NULL	NULL	%
1073	Mode 01	41-60	49	1	NULL	Accelerator pedal position D	2017-05-26 20:20	NULL	NULL	NULL
1074	Mode 01	41-60	4A	1	NULL	Accelerator pedal position E	2017-05-26 20:20	NULL	NULL	NULL
1075	Mode 01	41-60	4B	1	NULL	Accelerator pedal position F	2017-05-26 20:20	NULL	NULL	NULL
1076	Mode 01	41-60	4C	1	NULL	Commanded throttle actuator	2017-05-26 20:20	NULL	NULL	NULL
1077	Mode 01	41-60	4D	2	NULL	Time run with MIL on	2017-05-26 20:20	0	65535	NULL
1078	Mode 01	41-60	4E	2	NULL	Time since trouble codes cleared	2017-05-26 20:20	NULL	NULL	minutes
1079	Mode 01	41-60	4F	4	NULL	Maximum value for Fuel?Air equivalence ratio- oxygen sensor voltage- oxygen sensor current- and intake manifold absolute pressure	2017-05-26 20:20	0 0 0 0	255 255 255 2550	NULL
1080	Mode 01	41-60	50	4	NULL	Maximum value for air flow rate from mass air flow sensor	2017-05-26 20:20	0	2550	ratio V mA kPa

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1081	Mode 01	41-60	51	1	NULL	Fuel Type	2017-05-26 20:20	NULL	NULL	g/s
1082	Mode 01	41-60	52	1	NULL	Ethanol fuel %	2017-05-26 20:20	0	100	NULL
1083	Mode 01	41-60	53	2	NULL	Absolute Evap system Vapor Pressure	2017-05-26 20:20	0	327.68	%
1084	Mode 01	41-60	54	2	NULL	Evap system vapor pressure	2017-05-26 20:20	-32767	32768	kPa
1085	Mode 01	41-60	55	2	NULL	Short term secondary oxygen sensor trim- A: bank 1- B: bank 3	2017-05-26 20:20	-100	99.2	Pa
1086	Mode 01	41-60	56	2	NULL	Long term secondary oxygen sensor trim- A: bank 1- B: bank 3	2017-05-26 20:20	NULL	NULL	%
1087	Mode 01	41-60	57	2	NULL	Short term secondary oxygen sensor trim- A: bank 2- B: bank 4	2017-05-26 20:20	NULL	NULL	NULL
1088	Mode 01	41-60	58	2	NULL	Long term secondary oxygen sensor trim- A: bank 2- B: bank 4	2017-05-26 20:20	NULL	NULL	NULL
1089	Mode 01	41-60	59	2	NULL	Fuel rail absolute pressure	2017-05-26 20:20	0	655350	NULL
1090	Mode 01	41-60	5A	1	NULL	Relative accelerator pedal position	2017-05-26 20:20	0	100	kPa
1091	Mode 01	41-60	5B	1	NULL	Hybrid battery pack remaining life	2017-05-26 20:20	0	100	%
1092	Mode 01	41-60	5C	1	NULL	Engine oil temperature	2017-05-26 20:20	-40	210	%
1093	Mode 01	41-60	5D	2	NULL	Fuel injection timing	2017-05-26 20:20	-210	301.99	deg.C
1094	Mode 01	41-60	5E	2	NULL	Engine fuel rate	2017-05-26 20:20	0	3276.75	deg.
1095	Mode 01	41-60	5F	1	NULL	Emission requirements to which vehicle is designed	2017-05-26 20:20	NULL	NULL	L/h
1096	Mode 01	61-80	60	4	NULL	PIDs supported range 61-80	2017-05-26 20:20	NULL	NULL	NULL
1097	Mode 01	61-80	61	1	NULL	Driver's demand engine - percent torque	2017-05-26 20:20	-125	125	NULL

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1098	Mode 01	61-80	62	1	NULL	Actual engine - percent torque	2017-05-26 20:20	-125	125	%
1099	Mode 01	61-80	63	2	NULL	Engine reference torque	2017-05-26 20:20	0	65535	%
1100	Mode 01	61-80	64	5	NULL	Engine percent torque data	2017-05-26 20:20	-125	125	Nm
1101	Mode 01	61-80	65	2	NULL	Auxiliary input / output supported	2017-05-26 20:20	NULL	NULL	%
1102	Mode 01	61-80	66	5	NULL	Mass air flow sensor	2017-05-26 20:20	NULL	NULL	NULL
1103	Mode 01	61-80	67	3	NULL	Engine coolant temperature	2017-05-26 20:20	NULL	NULL	NULL
1104	Mode 01	61-80	68	7	NULL	Intake air temperature sensor	2017-05-26 20:20	NULL	NULL	NULL
1105	Mode 01	61-80	69	7	NULL	Commanded EGR and EGR Error	2017-05-26 20:20	NULL	NULL	NULL
1106	Mode 01	61-80	6A	5	NULL	Commanded Diesel intake air flow control and relative intake air flow position	2017-05-26 20:20	NULL	NULL	NULL
1107	Mode 01	61-80	6B	5	NULL	Exhaust gas recirculation temperature	2017-05-26 20:20	NULL	NULL	NULL
1108	Mode 01	61-80	6C	5	NULL	Commanded throttle actuator control and relative throttle position	2017-05-26 20:20	NULL	NULL	NULL
1109	Mode 01	61-80	6D	6	NULL	Fuel pressure control system	2017-05-26 20:20	NULL	NULL	NULL
1110	Mode 01	61-80	6E	5	NULL	Injection pressure control system	2017-05-26 20:20	NULL	NULL	NULL
1111	Mode 01	61-80	6F	3	NULL	Turbocharger compressor inlet pressure	2017-05-26 20:20	NULL	NULL	NULL
1112	Mode 01	61-80	70	9	NULL	Boost pressure control	2017-05-26 20:20	NULL	NULL	NULL
1113	Mode 01	61-80	71	5	NULL	Variable Geometry turbo (VGT) control	2017-05-26 20:20	NULL	NULL	NULL
1114	Mode 01	61-80	72	5	NULL	Wastegate control	2017-05-26 20:20	NULL	NULL	NULL
1115	Mode 01	61-80	73	5	NULL	Exhaust pressure	2017-05-26 20:20	NULL	NULL	NULL
1116	Mode 01	61-80	74	5	NULL	Turbocharger RPM	2017-05-26 20:20	NULL	NULL	NULL

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

ID	Mode	Segment	Code	Size	ReqInfo	Description	Date added	Min value	Max value	Units
1117	Mode 01	61-80	75	7	NULL	Turbocharger temperature	2017-05-26 20:20	NULL	NULL	NULL
1118	Mode 01	61-80	76	7	NULL	Turbocharger temperature	2017-05-26 20:20	NULL	NULL	NULL
1119	Mode 01	61-80	77	5	NULL	Charge air cooler temperature (CACT)	2017-05-26 20:20	NULL	NULL	NULL
1120	Mode 01	61-80	78	9	NULL	Exhaust Gas temperature (EGT) Bank 1	2017-05-26 20:20	NULL	NULL	NULL
1121	Mode 01	61-80	79	9	NULL	Exhaust Gas temperature (EGT) Bank 2	2017-05-26 20:20	NULL	NULL	NULL
1122	Mode 01	61-80	7A	7	NULL	Diesel particulate filter (DPF)	2017-05-26 20:20	NULL	NULL	NULL
1123	Mode 01	61-80	7B	7	NULL	Diesel particulate filter (DPF)	2017-05-26 20:20	NULL	NULL	NULL
1124	Mode 01	61-80	7C	9	NULL	Diesel Particulate filter (DPF) temperature	2017-05-26 20:20	NULL	NULL	NULL
1125	Mode 01	61-80	7D	1	NULL	NOx NTE (Not-To-Exceed) control area status	2017-05-26 20:20	NULL	NULL	NULL
1126	Mode 01	61-80	7E	1	NULL	PM NTE (Not-To-Exceed) control area status	2017-05-26 20:20	NULL	NULL	NULL
1127	Mode 01	61-80	7F	13	NULL	Engine run time	2017-05-26 20:20	NULL	NULL	NULL
1128	Mode 01	61-80	80	NULL	NULL	PIDs supported range 81-A0	2017-05-26 20:20	NULL	NULL	NULL

Extras din fluxul de telmetrie

TimeStamp	StrValue	CarID	Size	CodeID	Mode	Segment	Code	Units
20:37.6	0xec	1	1	1029	Mode 01	00-20	1D	NULL
20:37.6	0x6ad607cf	3	4	1080	Mode 01	41-60	50	ratio V mA kPa
20:37.6	0x31fda96c8f	2	5	1114	Mode 01	61-80	72	NULL
20:37.7	0x355237dc	2	4	1042	Mode 01	21-40	2A	NULL

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

TimeStamp	StrValue	CarID	Size	CodeID	Mode	Segment	Code	Units
20:37.7	0xd6f6	2	2	1063	Mode 01	21-40	3F	NULL
20:37.7	0x293f	3	2	1094	Mode 01	41-60	5E	deg.
20:37.7	0x1ef9d705	2	4	1054	Mode 01	21-40	36	NULL
20:37.7	0xdc0e	4	2	1002	Mode 01	00-20	2	NULL
20:37.7	0x9b73d6ab	3	4	1079	Mode 01	41-60	4F	NULL
20:37.7	0xad	3	1	1030	Mode 01	00-20	1E	NULL
20:39.4	0x4dd	3	2	1089	Mode 01	41-60	59	NULL
20:39.5	0xa9	3	1	1008	Mode 01	00-20	8	NULL
20:39.5	0xab12	4	2	1050	Mode 01	21-40	32	km
20:39.5	0xaa	3	1	1009	Mode 01	00-20	9	NULL
20:39.5	0xac	4	1	1069	Mode 01	41-60	45	ratio
20:39.5	0xe16f	1	2	1060	Mode 01	21-40	3C	NULL
20:39.5	0x7ddc1988	1	4	1039	Mode 01	21-40	27	NULL
20:39.5	0x21	2	1	1004	Mode 01	00-20	4	%
20:39.5	0xdfa5	3	2	1022	Mode 01	00-20	16	NULL
20:39.5	0x84f98371	1	4	1079	Mode 01	41-60	4F	NULL
20:40.5	0xd0a36c1	2	4	1036	Mode 01	21-40	24	kPa
20:40.6	0x95a4	2	2	1034	Mode 01	21-40	22	km
20:40.6	0xa6a8657aab82ef	2	7	1123	Mode 01	61-80	7B	NULL
20:40.6	0x6d151428	3	4	1040	Mode 01	21-40	28	NULL
20:40.6	0x3dd4413d	3	4	1058	Mode 01	21-40	3A	NULL
20:40.6	0x3dddf749	4	4	1065	Mode 01	41-60	41	NULL
20:40.6	0xb532e8e1ae5cf2	1	7	1105	Mode 01	61-80	69	NULL
20:40.6	0xa2b4392d	3	4	1058	Mode 01	21-40	3A	NULL
20:40.6	0xdef8	3	2	1021	Mode 01	00-20	15	volts / %
20:40.6	0xccf5160b	1	4	1001	Mode 01	00-20	1	NULL
20:42.8	0xad39cd97	1	4	1059	Mode 01	21-40	3B	NULL

TimeStamp	StrValue	CarID	Size	CodeID	Mode	Segment	Code	Units
20:42.9	0xc40b	4	2	1087	Mode 01	41-60	57	NULL
20:42.9	0x484070a269	1	5	1107	Mode 01	61-80	6B	NULL
20:42.9	0xdb67	1	2	1087	Mode 01	41-60	57	NULL
20:42.9	0x7333	4	2	1003	Mode 01	00-20	3	NULL
20:42.9	0xe6e022ab	4	4	1056	Mode 01	21-40	38	NULL
20:42.9	0xe1	4	1	1044	Mode 01	21-40	2C	NULL
20:42.9	0x30	4	1	1092	Mode 01	41-60	5C	%
20:42.9	0xc8457b6a	4	4	1037	Mode 01	21-40	25	ratio / V
20:42.9	0xd3	4	1	1009	Mode 01	00-20	9	NULL
20:43.9	0x5abf2e3c	2	4	1039	Mode 01	21-40	27	NULL
20:43.9	0x5f124e1f	1	4	1000	Mode 01	00-20	0	NULL
20:43.9	0xe20a1583	2	4	1059	Mode 01	21-40	3B	NULL
20:44.0	0x92	1	1	1090	Mode 01	41-60	5A	kPa
20:44.0	0x884e	4	2	1033	Mode 01	21-40	21	NULL
20:44.0	0xa21e	3	2	1088	Mode 01	41-60	58	NULL
20:44.0	0xac18	1	2	1023	Mode 01	00-20	17	NULL
20:44.0	0xd8cd7940	3	4	1054	Mode 01	21-40	36	NULL
20:44.0	0xaca7ba74d7e210	3	7	1122	Mode 01	61-80	7A	NULL
20:44.0	0x685e7860ab	3	5	1116	Mode 01	61-80	74	NULL
44:34.8	0xdf	3	1	1029	Mode 01	00-20	1D	NULL

Sistemul de analiza/calcul a datelor OBD2

In vederea traducerii informatiilor telemetrice obtinute de la computerele autovehiculelor se foloseste o harta de translatare a datelor de intrare primare. In acest sens datele codate pe octeti multiplii (de obicei 1-4 octeti respectiv 8-32 biti) sunt filtrate printr-un calcul in baza unei formule configurate pentru fiecare cod individual. Acest lucru se poate observa

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

cel mai simplu in exemplul/extrasul de date ale codurilor de telemetrie de mai jos unde practic ecuatia simpla care defineste valoarea reala de telemetrie este:

```
[VAL] = [READ] * [Mult] + [Add]
```

Unde `[READ]` este valoarea de intrare generata de computerele de bord ale autohevicolului.

Min value	Max value	Units	Enabled	Mult	Add
			0	1	0
			0	1	0
			0	1	0
			0	1	0
0.00	100.00	%	1	0.392156863	0
-40.00	215.00	deg.C	1	1	-40
-100.00	99.20	%	1	0.78125	-100
-100.00	99.20	%	1	0.78125	-99
-100.00	99.20	%	1	0.78125	-99
-100.00	99.20	%	1	0.78125	-99
0.00	765.00	kPa	1	3	0
0.00	255.00	kPa	1	1	0
0.00	16383.75	rpm	1	0.25	0
0.00	255.00	km/h	1	1	0
-64.00	63.50	deg. before TDC	0	0.5	-64
-40.00	215.00	dec.C	1	1	-40
0.00	655.35	grams/sec	1	0.01	0
0.00	100.00	%	1	0.392156863	0
			0		
			0		
-100.00	99.20	volt/% (1/200*A / 100/128*B-100)	0		
-100.00	99.20	volt/% (1/200*A /	0		

-100.00	99.20	100/128*B-100) volt/% (1/200*A / 100/128*B-100)	0

Codul sursa al sistemului

Cod sursa motor date Azure

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 25 19:43:09 2017
"""

from __future__ import print_function
import pandas as pd
import pyodbc
import urllib
import json
from sqlalchemy import create_engine
import datetime
import time as tm
import os

__author__      = "Andrei Ionut DAMIAN"
__copyright__   = "Copyright 2007 4E Software"
__credits__     = ["Andrei Simion"]
__license__     = "GPL"
__version__     = "1.3.1"
__maintainer__  = "Andrei Ionut DAMIAN"
__email__       = "damian@4esoft.ro"
__status__      = "Production"
__library__     = "AZURE SQL HELPER"
__created__     = "2017-01-25"
__modified__    = "2017-05-25"
```

```
__lib__         = "SQLHLP"



def start_timer():
    return tm.time()

def end_timer(start_timer):
    return(tm.time()-start_timer)

def print_progress(str_text):
    print("\r"+str_text, end='\r', flush=True)
    return

class MSSQLHelper:
  def __init__(self, config_file = "sql_config.txt", parent_log = None):

      self.DEBUG = 1
      self.debug_str_size = 35



      self.parent_log = parent_log
      self.MODULE = '[{} v{}]'.format(__library__,__version__)
      self._logger("INIT "+self.MODULE)
      cfg_file = open(config_file)
      config_data = json.load(cfg_file)
      cfg_file.close()
      self.driver   = config_data["driver"]
      self.server   = config_data["server" ]
      self.database = config_data["database"]
      self.username = config_data["username"]
      self.password = config_data["password"]
      self.cwd = os.getcwd()

      try:
          self.dfolder  = config_data["datafolder"]
      except:
          self.dfolder = "save"
          self.dfolder = os.path.join(self.cwd,self.dfolder)
```

```
        self.data_folder = self.dfolder
        self.dfolder = os.path.join(self.dfolder,"db_cache")

        if not os.path.isdir(self.dfolder):
            self._logger("Creating data folder:{}".format(
                                self.dfolder[-self.debug_str_size:]))
            os.makedirs(self.dfolder)
        else:
            self._logger("Using data folder:...{}".format(
                    self.dfolder[-self.debug_str_size:]))

        self.connstr = 'DRIVER=' + self.driver
        self.connstr+= ';SERVER=' + self.server
        self.connstr+= ';DATABASE=' + self.database
        self.connstr+= ';UID=' + self.username
        self.connstr+= ';PWD=' + self.password
        self.engine = None


        sql_params = urllib.parse.quote_plus(self.connstr)

        try:
            self._logger("ODBC Conn:
{}...".format(self.connstr[:self.debug_str_size]))
            self.conn = pyodbc.connect(self.connstr,
                                       timeout = 2)
            self.engine = create_engine("mssql+pyodbc:///?odbc_connect=%s"
% sql_params,
                                        connect_args={'connect_timeout':
2})
            self._logger("Connection created on "+self.server)
        except Exception as err: #pyodbc.Error as err:
            self._logger("FAILED ODBC Conn!")
            self.HandleError(err)
        return


    def Select(self,str_select, caching = True, convert_ascii = None):
        df = None
```

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
        try:
            str_fn = "".join(["_" if x in " ,;()*\\\\/[].><" else x for x
in str_select])
            str_fn = str_fn.replace("__","_").replace("__","_")
            str_fn += ".csv"
            str_fn = os.path.join(self.dfolder,str_fn)
            if self.DEBUG>1:
                self._logger("Using datafile: {}".format(str_fn))
            t0 = tm.time()
            if (not os.path.isfile(str_fn)) or (not caching):
                fmt_sql = " ".join(str_select.split())[:80]
                if self.DEBUG>0:
                    self._logger("Downloading data [{}..]
...".format(fmt_sql[:30]))
                else:
                    self._logger("Downloading data ...")
                df = pd.read_sql(str_select, self.conn)
                if convert_ascii != None:
                    # now convert columns to ascii
                    for col in convert_ascii:
                        df[col] = df[col].apply(lambda x: ''.join(
                                [" " if ord(i) < 32 or ord(i) > 126 else i
                                     for i in x]))
                if caching:
                    if self.DEBUG>0:
                        self._logger("Saving to [..{}]...".format(str_fn[-
self.debug_str_size:]))
                    else:
                        self._logger("Saving cache...")
                    df.to_csv(str_fn, index = False)
            else:
                if self.DEBUG>0:
                    self._logger("Loading file [..{}] ...".format(str_fn[-
self.debug_str_size:]))
                else:
                    self._logger("Loading file ...")
                df = pd.read_csv(str_fn)
            nsize = self.GetSize(df) / float(1024*1024)
            t1 = tm.time()
```

```
            tsec = t1-t0
            tmin = float(tsec) / 60
            self._logger("Dataset loaded: {:.2f}MB in {:.1f}s({:.1f}m) {}
rows".format(
                        nsize,
                        tsec,
                        tmin,
                        df.shape[0],
                        str_select))
            if self.DEBUG>1:
                self._logger("Dataset head(3):\n{}".format(df.head(2)))
            #self._logger("  READ TABLE time: {:.1f}s
({:.2f}min)".format(tsec,tmin))
        except Exception as err: #pyodbc.Error as err:
            self.HandleError(err)
        return df


    def ReadTable(self, str_table):
      str_select = "SELECT * FROM ["+str_table+"]"
      return self.Select(str_select)

    def GetEmptyTable(self, str_table):
      str_select = "SELECT TOP (1) * FROM ["+str_table+"]"
      return self.Select(str_select)[0:0]


    def ExecInsert(self, sInsertQuery):
        try:
            t0 = tm.time()
            cursor = self.conn
            cursor.execute(sInsertQuery)
            self.conn.commit()
            t1 = tm.time()
            tsec = t1-t0
            tmin = float(tsec) / 60
            self._logger("EXEC SQL  time: {:.1f}s
({:.2f}min)".format(tsec,tmin))
        except Exception as err: #pyodbc.Error as err:
```

```
            self.HandleError(err)
        return


  def SaveTable(self, df, sTable):
    dfsize = self.GetSize(df) / (1024*1024)
    try:
        self._logger("SAVING TABLE [APPEND]({:,} records
{:,.2f}MB)...".format(
                      df.shape[0],
                      dfsize))
        t0 = tm.time()
        df.to_sql(sTable,
                  self.engine,
                  index = False,
                  if_exists = 'append')
        t1 = tm.time()
        tsec = t1-t0
        tmin = float(tsec) / 60
        self._logger("DONE SAVE TABLE. Time = {:.1f}s
({:.2f}min)".format(tsec,tmin))
    except Exception as err: #pyodbc.Error as err:
        self.HandleError(err)
    return

  def OverwriteTable(self, df, sTable):
    dfsize = self.GetSize(df) / (1024*1024)
    try:
        self._logger("SAVING TABLE [OVERWRITE]({:,} records
{:,.2f}MB)...".format(
                      df.shape[0],
                      dfsize))
        t0 = tm.time()
        df.to_sql(sTable,
                  self.engine,
                  index = False,
                  if_exists = 'replace')
        t1 = tm.time()
        tsec = t1-t0
```

```
        tmin = float(tsec) / 60
        self._logger("DONE SAVE TABLE. Time = {:.1f}s
({:.2f}min)".format(tsec,tmin))
    except Exception as err: #pyodbc.Error as err:
        self.HandleError(err)
    return

  def Close(self):
    self.conn.close()
    return


  def HandleError(self, err):
      strerr = "ERROR: "+ str(err) #[:50]
      self._logger(strerr)
      return

  def GetSize(self,df):
      dfsize = df.values.nbytes + df.index.nbytes + df.columns.nbytes
      return dfsize


  def _logger(self, logstr, show = True):

      if self.parent_log != None:
          logstr = "[{}] ".format(__lib__) + logstr
          self.parent_log._logger(logstr,show)
      else:
          if not hasattr(self, 'log'):
              self.log = list()
          nowtime = datetime.datetime.now()
          strnowtime = nowtime.strftime("[{}][%Y-%m-%d %H:%M:%S]
".format(__lib__))
          logstr = strnowtime + logstr
          self.log.append(logstr)
          if show:
              print(logstr, flush = True)
      return
```

```
  def __exiT__(self, exc_type, exc_val, exc_tb):
      self.conn.close()
      self._logger("__exit__")
      return





if __name__ == '__main__':

    print("ERROR: MSSQLHelper is library only!")
```

Cod sursa modul central

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 26 17:45:41 2017

@author: Andrei
"""



__author__      = "Andrei Ionut DAMIAN"
__project__     = "GoDriveCarBox"
__copyright__   = "Copyright 2007 4E Software"
__credits__     = ["Andrei Simion"]
__license__     = "GPL"
__version__     = "0.1.1"
__maintainer__  = "Andrei Ionut DAMIAN"
__email__       = "damian@4esoft.ro"
__status__      = "Production"
__library__     = "DATA EXPLORER"
```

```
__created__     = "2017-01-25"
__modified__    = "2017-05-25"
__lib__         = "GDCBDE"


from gdcb_azure_helper import MSSQLHelper
import pandas as pd
from datetime import datetime as dt
import numpy as np
import os
import json

import time as tm


def clean_nonascii_df(df):
  for col in df.columns:
    if df[col].dtype=='O':
      df[col] = df[col].astype(str)
      df[col] = df[col].apply(
        lambda x: ''.join([" " if ord(i) < 32 or ord(i) > 126 else i
                           for i in x]))
  return df

class GDCBExplorer:
  """
  GDCB Data Explorer main class
   - uploads data to Azure via GDCB Azure Helper engine
   - downloads data for model training and prediction
   - acts as a general data broker
  """
  def __init__(self):
    self.FULL_DEBUG = True
    pd.options.display.float_format = '{:,.3f}'.format
    pd.set_option('expand_frame_repr', False)
    np.set_printoptions(precision = 3, suppress = True)


    self.MODULE = "{} v{}".format(__library__,__version__)
```

```
    self.s_prefix = dt.strftime(dt.now(),'%Y%m%d')
    self.s_prefix+= "_"
    self.s_prefix+=dt.strftime(dt.now(),'%H%M')
    self.s_prefix+= "_"
    self.cwd = os.getcwd()
    self.save_folder = os.path.join(self.cwd,"temp")
    self.log_file = os.path.join(self.save_folder,
                                 self.s_prefix + __lib__+"_log.txt")
    nowtime = dt.now()
    strnowtime = nowtime.strftime("[{}][%Y-%m-%d %H:%M:%S]
".format(__lib__))
    print(strnowtime+"Init log: {}".format(self.log_file))

    if not os.path.exists(self.save_folder):
        print(strnowtime+"CREATED TEMP LOG FOLDER:
{}".format(self.save_folder))
        os.makedirs(self.save_folder)
    else:
        print(strnowtime+"TEMP LOG FOLDER: {}".format(self.save_folder))
    self.sql_eng = MSSQLHelper(parent_log = self)
    self.setup_folder()
    self._logger("Work folder: [{}]".format(self.save_folder))



    self._logger("INIT "+self.MODULE)

    if self.FULL_DEBUG:
        self._logger(self.s_prefix)
        self._logger("__name__: {}".format(__name__))
        self._logger("__file__: {}".format(__file__))
    self._load_config()

    self.SetupVariables()
    return

  def _logger(self, logstr, show = True):
    """
    log processing method
    """
```

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
    if not hasattr(self, 'log'):
        self.log = list()
    nowtime = dt.now()
    strnowtime = nowtime.strftime("[{}][%Y-%m-%d %H:%M:%S]
".format(__lib__))
    logstr = strnowtime + logstr
    self.log.append(logstr)
    if show:
        print(logstr, flush = True)
    try:
        log_output = open(self.log_file, 'w')
        for log_item in self.log:
          log_output.write("%s\n" % log_item)
        log_output.close()
    except:
        print(strnowtime+"Log write error !", flush = True)
    return


  def setup_folder(self):
    """
    Setup folders for app
    """
    self.s_prefix = dt.strftime(dt.now(),'%Y%m%d')
    self.s_prefix+= "_"
    self.s_prefix+=dt.strftime(dt.now(),'%H%M')
    self.s_prefix+= "_"
    self.save_folder = self.sql_eng.data_folder
    self.out_file = os.path.join(self.save_folder,
                                 self.s_prefix +
__lib__+"_result_data.csv")
    self.log_file = os.path.join(self.save_folder,
                                 self.s_prefix + __lib__+"_log.txt")
    self._logger("LOGfile: {}".format(self.log_file[:30]))
    return

  def _load_config(self, str_file = 'gdcb_config.txt'):
    """
    Load JSON configuration file
```

```
    """

    cfg_file = open(str_file)
    self.config_data = json.load(cfg_file)
    return

  def SetupVariables(self):
    """
     load predictor variables from SQL Server repository and prepare raw-
data
     dataframe structure (by loading)
    """
    self._logger("Setup predictors and raw data repo...")
    s_pred_table = self.config_data["PREDICTOR_TABLE"]
    s_rawd_table = self.config_data["RAWDATA_TABLE"]
    s_cars_table = self.config_data["CARS_TABLE"]

    self.code_field = self.config_data["CODE_FIELD"]
    self.size_field = self.config_data["SIZE_FIELD"]

    self.raw_nval_field = self.config_data["RAW_NVAL_FIELD"]
    self.raw_sval_field = self.config_data["RAW_SVAL_FIELD"]
    self.raw_code_field = self.config_data["RAW_CODE_FIELD"]
    self.raw_time_field = self.config_data["RAW_TIME_FIELD"]
    self.raw_cari_field = self.config_data["RAW_CARI_FIELD"]

    self.df_predictors = self.sql_eng.ReadTable(s_pred_table)
    if not self.df_predictors is None:
      self.df_predictors.fillna(0,inplace = True)
      self._logger("Loaded {}
predictors".format(self.df_predictors.shape[0]))

    self.df_rawdata = self.sql_eng.GetEmptyTable(s_rawd_table)
    if not self.df_rawdata is None:
      self.df_rawdata.drop(self.config_data["RAW_IGNR_FIELD"],
                           axis=1, inplace=True)
      self._logger("RawData: {}".format(list(self.df_rawdata.columns)))

    self.df_cars = self.sql_eng.ReadTable(s_cars_table)
```

```
    if not self.df_cars is None:
      self._logger("Loaded {} cars".format(self.df_cars.shape[0]))

    self._logger("Done data preparation.")
    return

  def DumpRawData(self):
    """
    saves raw data to the sql table
    """
    assert not (self.sql_eng.engine is None)
    self._logger("Saving raw data ...")
    self.sql_eng.SaveTable(self.df_rawdata,
self.config_data["RAWDATA_TABLE"])
    self._logger("Done saving raw data.")
    return

  def EmptyRawData(self):
    self.df_rawdata = self.df_rawdata[0:0]
    return

  def _sample_number(self,nbytes):
    v = 0
    for i in range(nbytes*2):
      v += np.random.randint(0,16) * (16**i)
    return v

  def SampleRaw(self, sample_size):
    self._logger("Sampling data [{}]...".format(sample_size))
    nr_codes = self.df_predictors.shape[0]
    self.EmptyRawData()
    assert nr_codes != 0
    for i in range(sample_size):
      n = np.random.randint(0,nr_codes)
      c = np.random.randint(0, self.df_cars.shape[0])
      carid = self.df_cars.iloc[c,0]
      s_code = self.df_predictors.loc[n,self.code_field]
      nowtime = dt.now()
      strnowtime = nowtime.strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
```

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
      nbytes = int(self.df_predictors.loc[n,self.size_field])
      nval = self._sample_number(nbytes)
      sb16val = hex(nval)
      if nbytes>8:
        nval = 0
      self.df_rawdata.loc[i,self.raw_code_field] = s_code
      self.df_rawdata.loc[i,self.raw_nval_field] = nval
      self.df_rawdata.loc[i,self.raw_sval_field] = sb16val
      self.df_rawdata.loc[i,self.raw_time_field] = strnowtime
      self.df_rawdata.loc[i,self.raw_cari_field] = carid

    self.df_rawdata[self.raw_code_field] =
self.df_rawdata[self.raw_code_field].astype(int)
    self._logger("Done sampling data.")
    self.DumpRawData()


  def SampleRange(self, nr_samples, sample_size):
    self._logger("Sampling {} data of size [{}]...".format(nr_samples,
sample_size))
    t0 = tm.time()
    for i in range(nr_samples):
      self._logger("Sampling {}/{}".format(i,nr_samples))
      self.SampleRaw(sample_size)
    t1 = tm.time()
    self._logger("Data sampling for {} data of size [{}] finished in
{:.1f}s".format(
                    nr_samples, sample_size, t1-t0))
    return




if __name__ =="__main__":

  RUN_UPLOAD = False

  explorer = GDCBExplorer()
  if RUN_UPLOAD:
```

```
    dft = pd.read_csv("../tests/mode01_codes_raw.csv",encoding = "ISO-
8859-1")
    dft = clean_nonascii_df(dft)
    dft.to_csv("../tests/mode01_codes.csv", index=False)
    df = pd.read_csv("../tests/mode01_codes.csv")

    explorer.sql_eng.SaveTable(df,"Codes")



  explorer.SampleRange(100,100)
```

Cod sursa modul Machine Learning v2

```
//
// GoDriveCarBox Machine Learning Engine
//
// Created:            2017-01-30
// Last modified:      2017-05-30
//
// @copyright: 4E SOFTWARE SRL
//

#pragma once

#include "stdafx.h"
#include "stdio.h"
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <set>

#include <Eigen/Dense>
#include <Eigen/Core>
#include <Eigen/SVD>

#include <sys/stat.h>

#include <chrono>

#include <algorithm>
#include <random>

using namespace std;
using namespace std::chrono;


using Eigen::MatrixXd;
using Eigen::VectorXd;

using namespace Eigen;
using namespace std;
```

```
struct TrainCrossSplits
{
        MatrixXd X_train;
        MatrixXd X_cross;
        vector <string> Labels;
        VectorXd y_train;
        VectorXd y_cross;
};



class GenericEngine
{
private:
        long LoadedDataNrFields;
        long LoadedDataNrRows;
        long TrainTestSplitPos;

        std::default_random_engine random_engine;

protected:
        milliseconds start_time;
        milliseconds end_time;

        bool bBiasAdded; // variable that stores bias information for pre-loaded
data

        string CLF_NAME;
        long NR_FEATS;
        long NR_CLASSES;

public:

        bool VERBOSE_ENGINE;

        MatrixXd *X_loaded;
        VectorXd *y_loaded;

        MatrixXd *X_train;
        VectorXd *y_train;

        MatrixXd *X_cross;
        VectorXd *y_cross;

        MatrixXd *LoadedData;
        vector <string> LoadedDataHeader;
        vector <string> LabelsVector;


        GenericEngine()
        {
                CLF_NAME = "Generic Engine";

                // obtain a time-based seed:
                unsigned seed =
std::chrono::system_clock::now().time_since_epoch().count();
                random_engine = default_random_engine(seed);

                VERBOSE_ENGINE = true;

                bBiasAdded = false;

                X_train = NULL;
                y_train = NULL;
```

```
        X_loaded = NULL;
        y_loaded = NULL;
        X_cross = NULL;
        y_cross = NULL;
        LoadedData = NULL;
}

bool file_exists(const std::string& name);


void debug_info(string str_message)
{
        if (VERBOSE_ENGINE)
                printf("\n[DEBUG] %s", str_message.c_str());
}

void debug_info(string msg, MatrixXd mat)
{
        std::stringstream ss;
        ss << mat;
        string str_matrix = ss.str();
        string msgp = "\n[DEBUG] " + msg + "\n";
        printf(msgp.c_str());
        std::cout << str_matrix << std::endl;
}


void debug_info(MatrixXd mat)
{
        std::stringstream ss;
        ss << mat;
        string str_matrix = ss.str();
        printf("\n[DEBUG] Matrix:\n");
        std::cout << str_matrix << std::endl;
}

void debug_info(VectorXd vec, bool bHorizontal)
{
        std::stringstream ss;
        if (bHorizontal)
                ss << vec.transpose();
        else
                ss << vec;
        string str_vector = ss.str();
        printf("\n[DEBUG] Vector:\n");
        std::cout << str_vector << std::endl;
}

void debug_info(string msg, VectorXd vec, bool bHorizontal)
{
        std::stringstream ss;
        if (bHorizontal)
                ss << vec.transpose();
        else
                ss << vec;
        string str_vector = ss.str();
        string msgp = "\n[DEBUG] " + msg + "\n";
        printf(msgp.c_str());
        std::cout << str_vector << std::endl;
}
void debug_info()
{

        printf("\n[DEBUG] [PRESS ENTER]");
        int c = getc(stdin);
```

```
        }


        MatrixXd ShuffleMatrixRows(MatrixXd DataMatrix);


        int FindLabelId(vector <string> labels, string value);

        void BeginTimer();
        long EndTimer();

        vector <string> ToLabels(VectorXd y);

        TrainCrossSplits LoadCSV(const string& inputfile, const bool bShuffle =
false, const bool bAddBias = false);

};

class GenericLinearEngine : public GenericEngine
{
protected:

        long nr_batches; // how many batches have been processed (epochs, online
trainings, etc)
        VectorXd *SingleClassTheta;
        MatrixXd *Theta;
        VectorXd *J_values;

        void add_cost(double J);


private:

        void init();

public:



        GenericLinearEngine()
        {
                CLF_NAME = "VIRTUAL Generic Linear Engine";
                init();

        }
        ~GenericLinearEngine()
        {
                debug_info("Deleting object [" + CLF_NAME + "]");

                if (LoadedData != NULL)
                        delete LoadedData;
                if (X_loaded != NULL)
                        delete X_loaded;
                if (y_loaded != NULL)
                        delete y_loaded;
                if (X_train != NULL)
                        delete X_train;
                if (y_train != NULL)
                        delete y_train;
                if (X_cross != NULL)
                        delete X_cross;
                if (y_cross != NULL)
                        delete y_cross;
                if (Theta != NULL)
                        delete Theta;
                if (SingleClassTheta != NULL)
```

```
                        delete SingleClassTheta;
                if (J_values != NULL)
                        delete J_values;
        }

        VectorXd PredictSingleClass(MatrixXd X);

        virtual MatrixXd Predict(MatrixXd X);

        vector <string> PredictLabels(MatrixXd X);

        vector <string> PredictLabelsUsingYHat(MatrixXd y_hat);

        string GetName();
        MatrixXd& GetTheta();

        float NRMSE(VectorXd y_hat, VectorXd y);
        float RMSE(VectorXd y_hat, VectorXd y);
        float CrossEvaluationSingleClass(bool bClass);
        float TrainEvaluationSingleClass(bool bClass);

        float CrossEvaluation(bool bClass);
        float TrainEvaluation(bool bClass);
};

class NormalRegressor : public GenericLinearEngine
{
protected:

        int t;

public:
        NormalRegressor()
        {
                NR_FEATS = 0;
                NR_CLASSES = 0;
                CLF_NAME = "Batch Normal Regressor";
        }

        void Train(MatrixXd X, MatrixXd y);
        void Train();

};

class OnlineClassifier :  public GenericLinearEngine
{
protected:

        // temp variables

        MatrixXd LastYHat;
        MatrixXd LastGrad;
        MatrixXd LastXObs;
        MatrixXd LastYOHM;
        MatrixXd LastYERR;

        double LearningRate;

        MatrixXd softmax(MatrixXd z);
        double cross_entropy(MatrixXd yOHM, MatrixXd y_hat);


public:
        OnlineClassifier(int nr_features, int nr_classes, vector <string> &labels,
double alpha_learning_rate)
        {
```

```
                CLF_NAME = "Online Linear Classifier";
                NR_FEATS = nr_features;
                NR_CLASSES = nr_classes;
                LabelsVector = labels;
                LearningRate = alpha_learning_rate;

                Theta = new MatrixXd(NR_FEATS+1, NR_CLASSES); // add 1 row for
biases

                Theta->fill(0);

        }

        void SimulateOnlineTrain();

        void OnlineTrain(MatrixXd xi, VectorXd yi);

        double CostFunction();

        MatrixXd Predict(MatrixXd X);




};



//
// BEGIN Generic Engine Class definitions - basic ancestor helper class
//

inline bool GenericEngine::file_exists(const std::string & name)
{
        if (FILE *file = fopen(name.c_str(), "r")) {
                fclose(file);
                return true;
        }
        else {
                return false;
        }
}
inline MatrixXd GenericEngine::ShuffleMatrixRows(MatrixXd DataMatrix)
{
        long size = DataMatrix.rows();
        PermutationMatrix<Dynamic, Dynamic> perm(size);
        perm.setIdentity();


        std::shuffle(perm.indices().data(),
                                perm.indices().data() + perm.indices().size(),
                                this->random_engine);

        MatrixXd A_perm = perm * DataMatrix; // permute rows
        return(A_perm);
}

inline int GenericEngine::FindLabelId(vector<string> labels, string value)
{
        int pos = find(labels.begin(), labels.end(), value) - labels.begin();

        if (pos >= labels.size()) {
                //old_name_ not found
                pos = -1;
        }
```

```
        return(pos);
}
void GenericEngine::BeginTimer()
{
        milliseconds ms = duration_cast< milliseconds >(
                system_clock::now().time_since_epoch()
                );
        start_time = ms;
}

inline long GenericEngine::EndTimer()
{
        milliseconds ms = duration_cast< milliseconds >(
                system_clock::now().time_since_epoch()
                );
        end_time = ms;
        return (end_time - start_time).count();
}

inline vector<string> GenericEngine::ToLabels(VectorXd y)
{
        vector <string> labels;
        for (long i = 0;i < y.size();i++)
        {
                string s = LabelsVector[y(i)];
                labels.push_back(s);
        }
        return(labels);
}

inline TrainCrossSplits GenericEngine::LoadCSV(const string & inputfile, const bool
bShuffle, const bool bAddBias)
{
        int nr_rows = 0;
        int nr_cols = 0;
        string fname = inputfile;
        TrainCrossSplits rec_results;

        if (!file_exists(inputfile))
                throw std::invalid_argument("Received invalid file in LoadCSV: " +
fname);

        ifstream infile(fname, std::ifstream::in);


        if (!infile.good())
                throw std::invalid_argument("Received invalid file in LoadCSV: " +
fname);

        debug_info("Loading " + fname + " dataset...");
        vector< vector<string> > result;
        while (!infile.eof())
        {
                //go through every line
                string line;

                getline(infile, line);

                vector <string> record;
                nr_cols = 0;

                std::size_t prev = 0, pos;
                while ((pos = line.find_first_of(",;", prev)) != std::string::npos)
                {
                        if (pos > prev)
                        {
```

```
                                record.push_back(line.substr(prev, pos - prev));
                                nr_cols++;
                        }

                        prev = pos + 1;
                }
                if (prev < line.length())
                {
                        record.push_back(line.substr(prev, std::string::npos));
                        nr_cols++;
                }


                if (nr_cols > 0)
                {
                        result.push_back(record);
                        nr_rows++;
                }
        }


        //
        // now load whole data, X and y matrices
        // assume last column of loaded data is the results / labels
        //

        LoadedDataNrFields = result[0].size();
        LoadedDataNrRows = nr_rows - 1; // rows minus field names row

        debug_info("Loaded " + std::to_string(LoadedDataNrRows) + " X " +
std::to_string(LoadedDataNrFields) + " dataset");

        LoadedData = new MatrixXd(LoadedDataNrRows, LoadedDataNrFields);
        y_loaded = new VectorXd(LoadedDataNrRows);
        X_loaded = new MatrixXd(LoadedDataNrRows, LoadedDataNrFields - 1);

        std::set <string> LabelsSet;


        long i, j;
        for (j = 0;j < LoadedDataNrFields;j++)
                LoadedDataHeader.push_back((string)result[0][j]);

        //
        // assume dataset is curated and ONLY last column contains text labels
        //

        vector <string> loaded_labels;

        for (i = 0;i < LoadedDataNrRows;i++)
                for (j = 0;j < LoadedDataNrFields;j++)
                {
                        double fcell = 0;
                        string scell = result[i + 1][j];
                        try
                        {
                                if (j != ((LoadedDataNrFields - 1)))
                                        fcell = ::atof(scell.c_str());
                        }
                        catch (...)
                        {

                        }
                        (*LoadedData)(i, j) = fcell;
                        if (j == (LoadedDataNrFields - 1))
                        {
```

Str Gheorghe Titeica nr 6, Sector 2,
Bucuresti
Romania

+4 031 4055287
+4 031 4012234
+4 0721 368 127

Website:
http://www.4esoft.com

email: office@4esoft.com

```
                                LabelsSet.insert(scell);
                                loaded_labels.push_back(scell);
                        }

                }

        LabelsVector.assign(LabelsSet.begin(), LabelsSet.end());

        for (int label_idx = 0;label_idx < loaded_labels.size();label_idx++)
        {
                string c_label = loaded_labels[label_idx];
                int iLabel = FindLabelId(LabelsVector, c_label);
                (*LoadedData)(label_idx, LoadedDataNrFields - 1) = iLabel;

        }


        if (bShuffle)
        {
                MatrixXd ttt = ShuffleMatrixRows(*LoadedData);
                *LoadedData = ttt;

        }

        float test_size = 0.2;
        int test_rows = LoadedDataNrRows * test_size;
        int train_rows = LoadedDataNrRows - test_rows;
        TrainTestSplitPos = train_rows;

        *X_loaded = LoadedData->leftCols(LoadedDataNrFields - 1);
        *y_loaded = LoadedData->rightCols(1);

        NR_FEATS = X_loaded->cols();
        NR_CLASSES = LabelsVector.size();


        if (bAddBias)
        {
                // now add bias
                VectorXd bias(LoadedDataNrRows);
                bias.fill(1);
                MatrixXd *TempX = new MatrixXd(LoadedDataNrRows, LoadedDataNrFields
- 1 + 1); // bias size
                *TempX << bias, *X_loaded;
                bBiasAdded = true;
                delete X_loaded;
                X_loaded = TempX;
                // done adding bias
        }

        X_train = new MatrixXd(X_loaded->topRows(train_rows));
        X_cross = new MatrixXd(X_loaded->bottomRows(test_rows));

        y_train = new VectorXd(y_loaded->head(train_rows));
        y_cross = new VectorXd(y_loaded->tail(test_rows));

        rec_results.X_cross = *X_cross;
        rec_results.X_train = *X_train;
        rec_results.y_cross = *y_cross;
        rec_results.y_train = *y_train;
        rec_results.Labels = LabelsVector;

        return(rec_results);

}
```

```
//
// END Generig Engine Class definitions
//

//
// BEGIN Generic Linear Engine (Virtual class)
//

inline float GenericLinearEngine::NRMSE(VectorXd y_hat, VectorXd y)
{
        float maxmin = y.maxCoeff()-y.minCoeff();
        return(RMSE(y_hat, y) / maxmin);
}

inline float GenericLinearEngine::RMSE(VectorXd y_hat, VectorXd y)
{
        long nr_obs = y.size();
        VectorXd errors = (y-y_hat);
        if (VERBOSE_ENGINE)
        {
                debug_info("Errors (last 3):");
                debug_info(errors.tail(3));
        }
        double sqNorm = errors.squaredNorm();
        return(sqrt(sqNorm / nr_obs));
}

inline void GenericLinearEngine::add_cost(double J)
{
        if (nr_batches == 0)
        {
                // first use :)
                J_values = new VectorXd(1);
                (*J_values)(nr_batches) = J;
        }
        else
        {
                J_values->conservativeResize(nr_batches + 1);
                (*J_values)(nr_batches) = J;
        }
        nr_batches++;
}

inline void GenericLinearEngine::init()
{

        debug_info("Generating object [" + CLF_NAME + "]");

        nr_batches = 0;
        Theta = NULL;
        SingleClassTheta = NULL;
        J_values = NULL;


}

double myexp(double val)
{
        return(exp(val));
}

MatrixXd& GenericLinearEngine::GetTheta()
{
        return *Theta;
}
string GenericLinearEngine::GetName()
```

```
{
        return(CLF_NAME);
}





double myround(double f)
{
        return(round(f));
}

inline VectorXd GenericLinearEngine::PredictSingleClass(MatrixXd X)
{
        VectorXd *pred = new VectorXd(X.rows());

        *pred = X * (*SingleClassTheta);

        return(*pred);
}

inline MatrixXd GenericLinearEngine::Predict(MatrixXd X)
{
        MatrixXd preds = X * (*Theta);
        return(preds);
}

inline vector<string> GenericLinearEngine::PredictLabels(MatrixXd X)
{
        MatrixXd y_hat = Predict(X);
        vector <string> PredictedLabels;
        for (long i = 0;i < y_hat.rows();i++)
        {
                int y_hat_idx;
                y_hat.row(i).maxCoeff(&y_hat_idx);
                PredictedLabels.push_back(LabelsVector[y_hat_idx]);
        }
        return(PredictedLabels);
}

inline vector<string> GenericLinearEngine::PredictLabelsUsingYHat(MatrixXd y_hat)
{
        vector <string> PredictedLabels;
        for (long i = 0;i < y_hat.rows();i++)
        {
                int y_hat_idx;
                y_hat.row(i).maxCoeff(&y_hat_idx);
                PredictedLabels.push_back(LabelsVector[y_hat_idx]);
        }
        return(PredictedLabels);
}


inline float GenericLinearEngine::TrainEvaluationSingleClass(bool bClass)
{
        double dResult = 0.0f;
        VectorXd y = *y_train;
        MatrixXd X = *X_train;
        long nr_train = y.size();
        if (SingleClassTheta == NULL && Theta == NULL)
                return (dResult);

        VectorXd y_hat = PredictSingleClass(X);
```

```
        long nr_obs = y_hat.size();

        if (VERBOSE_ENGINE)
        {
                debug_info("Train Y_Hat vs. Y_train (last 3)");
                MatrixXd result(nr_train, 2);
                result << y_hat, y;
                debug_info(result.bottomRows(3));
        }


        if (bClass)
        {
                VectorXd y_hat_Rounded = y_hat.unaryExpr(ptr_fun(myround));
                long positives = 0;
                for (long i = 0;i < nr_obs;i++)
                {
                        if (y_hat_Rounded(i) == (y)(i))
                                positives++;
                }
                dResult = (double)positives / nr_obs;
        }
        else
        {
                dResult = NRMSE(y_hat, y);
        }

        return (dResult);
}

inline float GenericLinearEngine::CrossEvaluation(bool bClass)
{
        double dResult = 0.0f;
        VectorXd y = *y_cross;
        MatrixXd X;
        if (!bBiasAdded)
                X = *X_cross;
        else
                X = X_cross->rightCols(NR_FEATS);

        long nr_cross = y.size();
        if (Theta == NULL)
                return (dResult);

        MatrixXd y_hat = Predict(X);

        long nr_obs = X.rows();

        if (VERBOSE_ENGINE)
        {
                MatrixXd result(nr_cross, y_hat.cols() + 1);
                result << y_hat, y;
                debug_info("Cross Y_Hat vs. Y_cross (last
5):",result.bottomRows(5));
        }


        if (bClass)
        {
                vector <string> preds = PredictLabelsUsingYHat(y_hat);
                long positives = 0;
                for (long i = 0;i < nr_obs;i++)
                {
                        string predicted = preds[i];
                        string label = LabelsVector[(int)y(i)];
                        if (predicted == label)
```

```
                                positives++;
                }
                dResult = (double)positives / nr_obs;
        }
        else
        {
                dResult = -1;
        }

        return (dResult);

}

inline float GenericLinearEngine::TrainEvaluation(bool bClass)
{
        double dResult = 0.0f;
        VectorXd y = *y_train;
        MatrixXd X;
        if (!bBiasAdded)
                X = *X_train;
        else
                X = X_train->rightCols(NR_FEATS);

        long nr_cross = y.size();
        if (Theta == NULL)
                return (dResult);

        MatrixXd y_hat = Predict(X);

        long nr_obs = X.rows();

        if (VERBOSE_ENGINE)
        {
                MatrixXd result(nr_cross, y_hat.cols() + 1);
                result << y_hat, y;
                debug_info("Train Y_Hat vs. Y_train (last 5):",
result.bottomRows(5));
        }


        if (bClass)
        {
                vector <string> preds = PredictLabelsUsingYHat(y_hat);
                long positives = 0;
                for (long i = 0;i < nr_obs;i++)
                {
                        string predicted = preds[i];
                        string label = LabelsVector[(int)y(i)];
                        if (predicted == label)
                                positives++;
                }
                dResult = (double)positives / nr_obs;
        }
        else
        {
                dResult = -1;
        }

        return (dResult);

}

inline float GenericLinearEngine::CrossEvaluationSingleClass(bool bClass)
{
        double dResult = 0.0f;
        VectorXd y = *y_cross;
```

```
        MatrixXd X = *X_cross;
        long nr_cross = y.size();
        if (SingleClassTheta == NULL && Theta == NULL)
                return (dResult);

        VectorXd y_hat = PredictSingleClass(X);
        long nr_obs = y_hat.size();

        if (VERBOSE_ENGINE)
        {
                debug_info("Cross Y_Hat vs. Y_cross (last 3)");
                MatrixXd result(nr_cross, 2);
                result << y_hat, y;
                debug_info(result.bottomRows(3));
        }


        if (bClass)
        {
                VectorXd y_hat_Rounded = y_hat.unaryExpr(ptr_fun(myround));
                long positives = 0;
                for (long i = 0;i < nr_obs;i++)
                {
                        if (y_hat_Rounded(i) == y(i))
                                positives++;
                }
                dResult = (double) positives / nr_obs;
        }
        else
        {
                dResult = NRMSE(y_hat, y);
        }

        return (dResult);
}
//
// END Generic Linear Engine virtual class
//



//
// BEGIN Normal Regressor class definitions
//
void NormalRegressor::Train(MatrixXd X, MatrixXd y)
{
        X_train = new MatrixXd(X);
        y_train = new VectorXd(y);
        Train();
}

template<typename _Matrix_Type_>
_Matrix_Type_ pseudoInverse(const _Matrix_Type_ &a, double epsilon =
std::numeric_limits<double>::epsilon())
{
        Eigen::JacobiSVD< _Matrix_Type_ > svd(a, Eigen::ComputeThinU |
Eigen::ComputeThinV);
        double tolerance = epsilon * std::max(a.cols(), a.rows())
*svd.singularValues().array().abs()(0);
        return svd.matrixV() *  (svd.singularValues().array().abs() >
tolerance).select(svd.singularValues().array().inverse(), 0).matrix().asDiagonal()
* svd.matrixU().adjoint();
}

void NormalRegressor::Train()
{
```

```
        debug_info("Training: " + CLF_NAME);
        MatrixXd X = *X_train;
        VectorXd y = *y_train;
        MatrixXd xTx = X.transpose() * X;
        MatrixXd xT = X.transpose();

        VectorXd TempTheta1(X.cols());
        VectorXd TempTheta2(X.cols());
        long duration1;
        long duration2;

        if (VERBOSE_ENGINE)
        {
                // 1st solving with pseudo-inverse
                high_resolution_clock::time_point t1 = high_resolution_clock::now();
                MatrixXd xTxInv = pseudoInverse(xTx);
                TempTheta1 = xTxInv * xT * y;
                high_resolution_clock::time_point t2 = high_resolution_clock::now();
                duration1 = duration_cast<microseconds>(t2 - t1).count();



                // now second method
                high_resolution_clock::time_point t3 = high_resolution_clock::now();
                TempTheta2 = xTx.ldlt().solve(xT * y);
                high_resolution_clock::time_point t4 = high_resolution_clock::now();
                duration2 = duration_cast<microseconds>(t4 - t3).count();

                //SingleClassTheta = new VectorXd(TempTheta1);
                SingleClassTheta = new VectorXd(TempTheta2);


        }
        else
        {
                // now second method
                TempTheta2 = xTx.ldlt().solve(xT * y);
                SingleClassTheta = new VectorXd(TempTheta2);
        }


        if (VERBOSE_ENGINE)
        {
                debug_info("X data features size = " + to_string(X_loaded->cols()));

                debug_info("Theta PInv = " + to_string(duration1) + " microsec");
                debug_info("Theta ldlt = " + to_string(duration2) + " microsec");

                debug_info("T1(pinv) T2(ldlt):");
                MatrixXd comp(TempTheta1.size(), 2);
                comp << TempTheta1, TempTheta2;
                debug_info(comp);
                if (*SingleClassTheta == TempTheta2)
                        debug_info("Using Theta2");
                else
                        debug_info("Using Theta1");

        }
}
//
// END Normal Regressor class definitions
//



inline void OnlineClassifier::SimulateOnlineTrain()
```

```
{
        if (Theta != NULL)
                delete Theta;
        Theta = new MatrixXd(NR_FEATS + 1, NR_CLASSES);
        Theta->fill(0); // reset Theta

        long TEST_DEBUG = 1000;

        BeginTimer();

        for (long i = 0;i < X_train->rows();i++)
        {
                MatrixXd obs = X_train->row(i);
                VectorXd yi(1);
                yi(0) = (*y_train)(i);


                if (VERBOSE_ENGINE)// && (i == TEST_DEBUG))
                {
                        std::stringstream ss;
                        for (size_t i = 0; i < yi.size(); ++i)
                        {
                                if (i != 0)
                                        ss << ",";
                                ss << yi[i];
                        }

                        debug_info("Training "+to_string(i)+" th example with y = " +
ss.str(),obs);
                }

                MatrixXd xi;
                if (bBiasAdded)
                        xi = obs.rightCols(NR_FEATS);
                else
                        xi = obs;

                OnlineTrain(xi, yi);

                if (VERBOSE_ENGINE)// && (i == TEST_DEBUG))
                {
                        //long time_cost = EndTimer();
                        //debug_info("Total time = " + to_string(time_cost) + " ms");
                        debug_info("y_OHM (1 row): ",LastYOHM.topRows(1));
                        debug_info("y_hat (1 row): ",LastYHat.topRows(1));
                        debug_info("error (1 row): ",LastYERR.topRows(1));
                        debug_info("Gradient (2 rows): ",LastGrad.topRows(2));

                        debug_info("J array las val: ",J_values->tail(1), true);
                        debug_info("Theta (2 rows): ",Theta->topRows(2));
                        //debug_info();
                }
        }
}

//
// BEGIN Online Classifier definitions
//
// yi is index in VectorLabels
void OnlineClassifier::OnlineTrain(MatrixXd xi, VectorXd yi)
{
        long nr_rows = xi.rows();
        long nr_cols = xi.cols();

        VectorXd bias(nr_rows);
        bias.fill(1);
```

```
        MatrixXd TempX(nr_rows, nr_cols + 1);
        TempX << bias, xi;

        long m = nr_rows; // for convenience
        MatrixXd yOHM(nr_rows, NR_CLASSES);
        yOHM.fill(0);
        for (long i = 0;i < nr_rows;i++)
        {
                for (long j = 0;j < NR_CLASSES;j++)
                        // now assume LabelsVector is correctly constructed
                        // and yi[i] is index in that vector
                        if (yi(i) == j)
                                yOHM(i, j) = 1;
        }

        // now we have the one hot matrix lets start working !
        MatrixXd y_hat = Predict(xi);
        double J = (1.0 / m) * cross_entropy(yOHM, y_hat); // MUST add
regularization
        add_cost(J);

        MatrixXd error = yOHM - y_hat;

        MatrixXd Grad = (-1.0 / m) * TempX.transpose() * error; // MUST add
regularization


        *Theta = *Theta - (LearningRate * Grad);

        LastGrad = Grad;
        LastYOHM = yOHM;
        LastYHat = y_hat;
        LastYERR = error;
        LastXObs = xi;

}

inline double OnlineClassifier::CostFunction()
{

        return 0.0;
}
inline MatrixXd OnlineClassifier::Predict(MatrixXd X)
{
        long nr_rows = X.rows();
        long nr_cols = X.cols();

        VectorXd bias(nr_rows);
        bias.fill(1);
        MatrixXd TempX(nr_rows, nr_cols +1);
        TempX << bias, X;
        MatrixXd XTheta = TempX * (*Theta);

        MatrixXd SM = softmax(XTheta);

        return(SM);
}



inline MatrixXd OnlineClassifier::softmax(MatrixXd z)
{
        MatrixXd SM(z.rows(), Theta->cols());

        ArrayXXd arr(z);
```

```
        // first shift values
        arr = arr - z.maxCoeff();
        arr = arr.exp();

        //cout << z;
        //cout << arr;

        ArrayXd sums = arr.rowwise().sum();

        arr.colwise() /= sums;

        SM = arr.matrix();


        return(SM);

}

double myclip(double val)
{
        double eps = 1e-15;
        if (val < eps)
                return(eps);
        else
                if (val > (1 - eps))
                        return(1 - eps);
                else
                        return(val);
}
inline double OnlineClassifier::cross_entropy(MatrixXd yOHM, MatrixXd y_hat)
{
        //y_hat = y_hat.unaryExpr(ptr_fun(myclip));

        MatrixXd J_matrix = (yOHM.array() * y_hat.array().log()).matrix();
        double J = -(J_matrix.sum());
        return(J);
}
//
// END Online Classifier definitions
//
```